

[illegible]

3

Sy

MT

MT
MT

MT

MT
MT

MT
MT

MT
MTMT
MTMT
MT

MT

MT

MT
MT

MT

MT
MT

MT
MTMT
MTMT
MT

MT

MT

MT
MT

MI

MT
MTMT
MTMT
MT

MT

M1
M1

M1

M1
M1M1
M1

100

1

1

100

10

1

★★

[illegible]

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```

(2) 56
(2) 70
(3) 191

HISTORY ; Detailed current edit history
DECLARATIONS
OTSS\$POWGG - G REAL*8 to G REAL*8 giving G REAL*8 result


```

0000 1      .TITLE OTSSPOWGG - G REAL*8 ** G REAL*8 power routine
0000 2      .IDENT /2-004/ ; File: OTSPOWGG.MAR EDIT: JCW2004
0000 3
0000 4
0000 5 *****
0000 6
0000 7      *
0000 8      * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9      * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10     * ALL RIGHTS RESERVED.
0000 11     *
0000 12     * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13     * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14     * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15     * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16     * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17     * TRANSFERRED.
0000 18     *
0000 19     * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20     * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21     * CORPORATION.
0000 22     *
0000 23     * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24     * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25     *
0000 26     *****
0000 27
0000 28
0000 29     FACILITY: Language support library - user callable
0000 30     ++
0000 31     ABSTRACT:
0000 32
0000 33         G double base to G double power.
0000 34
0000 35         Floating overflow can occur
0000 36         Undefined exponentiation can occur if:
0000 37             1) Negative base
0000 38             2) 0 base and power is 0 or negative.
0000 39
0000 40
0000 41
0000 42     --
0000 43
0000 44     VERSION: 2
0000 45
0000 46     HISTORY:
0000 47
0000 48     AUTHOR:
0000 49         Bob Hanek 8-Mar-83: Version 2
0000 50
0000 51     MODIFIED BY:
0000 52
0000 53
0000 54

```



```
0000 56 .SBTTL HISTORY ; Detailed current edit history
0000 57
0000 58
0000 59 : Edit history for Version 2 of OTSS$POWGG
0000 60 :
0000 61 : 2-001 - Implemented new algorithm. RNH 8-Mar-83
0000 62 : 2-002 - Change table INDEX to be local instead of GLOBAL. LEB 24-May-1983
0000 63 : 2-003 - Change instruction using INDEX(Rx) to INDEX[Rx] to avoid linker
0000 64 : errors. LEB 25-May-1983
0000 65 : 2-004 - Changed the ACMASK to include registers R9 and R10. This is needed
0000 66 : so that the entry masks of OTSS$POWGG and UVX$POWGG are identical.
0000 67 : JCW 17-JUL-84.
0000 68 :
0000 69
0000 70 .SBTTL DECLARATIONS
0000 71
0000 72 :
0000 73 : INCLUDE FILES:
0000 74 :
0000 75 :
0000 76 :
0000 77 : EXTERNAL SYMBOLS:
0000 78 :
0000 79 :
0000 80 .DSABL GBL
0000 81 .EXTRN MTH$$SIGNAL ; Error signal routine
0000 82 .EXTRN MTH$K_UNDEXP ; Undefined exponentiation code
0000 83 .EXTRN MTH$K_FLOUNDMAT ; Floating point underflow code
0000 84 .EXTRN MTH$K_FLOOVEMAT ; Floating point overflow code
0000 85 :
0000 86 :
0000 87 : MACROS:
0000 88 :
0000 89 : $SFDEF ; Define stack frame symbols
0000 90 :
0000 91 : EQUATED SYMBOLS:
0000 92 :
00000004 0000 93 : base = 4 ; base input formal - by-value
0000000C 0000 94 : exp = 12 ; exponent input formal - by-value
00000008 0000 95 : fexp = 8 ; exponent when base is floating
0000 96 : by-value
000007FC 0000 97 : ACMASK = ^M<R2, R3, R4, R5, R6, R7, R8, R9, R10>
0000 98 :
0000 99 :
0000 100 : OWN STORAGE: None
0000 101 :
0000 102 :
0000 103 :
0000 104 : PSECT DECLARATIONS:
0000 105 :
00000000 106 : .PSECT _OTSS$CODE PIC,SHR,LONG,EXE,NOWRT
0000 107 : ; program section for OTSS$ code
0000 108 :
0000 109 : Constants
0000 110 :
0000 111 :
0000 112 :
```


								0000	113	: The INDEX table gives the appropriate entry in the A_TABLE									
								0000	114	:									
								0000	115										
03	02	02	02	01	01	01	00	0000	116	INDEX:	.BYTE	^X00,	^X01,	^X01,	^X01,	^X02,	^X02,	^X02,	^X03
05	05	05	04	04	04	03	03	0008	117		.BYTE	^X03,	^X03,	^X04,	^X04,	^X04,	^X05,	^X05,	^X05
08	07	07	07	07	06	06	06	0010	118		.BYTE	^X06,	^X06,	^X06,	^X07,	^X07,	^X07,	^X07,	^X08
0A	0A	0A	09	09	09	08	08	0018	119		.BYTE	^X08,	^X08,	^X09,	^X09,	^X09,	^X0A,	^X0A,	^X0A
0C	0C	0C	0C	0B	0B	0B	0A	0020	120		.BYTE	^X0A,	^X0B,	^X0B,	^X0B,	^X0C,	^X0C,	^X0C,	^X0C
0F	0E	0E	0E	0E	0D	0D	0D	0028	121		.BYTE	^X0D,	^X0D,	^X0D,	^X0E,	^X0E,	^X0E,	^X0E,	^X0F
11	10	10	10	10	0F	0F	0F	0030	122		.BYTE	^X0F,	^X0F,	^X0F,	^X10,	^X10,	^X10,	^X10,	^X11
13	12	12	12	12	11	11	11	0038	123		.BYTE	^X11,	^X11,	^X11,	^X12,	^X12,	^X12,	^X12,	^X13
14	14	14	14	14	13	13	13	0040	124		.BYTE	^X13,	^X13,	^X13,	^X14,	^X14,	^X14,	^X14,	^X14
16	16	16	16	15	15	15	15	0048	125		.BYTE	^X15,	^X15,	^X15,	^X15,	^X16,	^X16,	^X16,	^X16
18	18	18	17	17	17	17	17	0050	126		.BYTE	^X17,	^X17,	^X17,	^X17,	^X17,	^X18,	^X18,	^X18
1A	1A	19	19	19	19	18	18	0058	127		.BYTE	^X18,	^X18,	^X19,	^X19,	^X19,	^X19,	^X1A,	^X1A
1B	1B	1B	1B	1B	1A	1A	1A	0060	128		.BYTE	^X1A,	^X1A,	^X1A,	^X1B,	^X1B,	^X1B,	^X1B,	^X1B
1D	1D	1D	1C	1C	1C	1C	1C	0068	129		.BYTE	^X1C,	^X1C,	^X1C,	^X1C,	^X1C,	^X1D,	^X1D,	^X1D
1E	1E	1E	1E	1E	1D	1D	1D	0070	130		.BYTE	^X1D,	^X1D,	^X1D,	^X1E,	^X1E,	^X1E,	^X1E,	^X1E
20	20	20	1F	1F	1F	1F	1F	0078	131		.BYTE	^X1F,	^X1F,	^X1F,	^X1F,	^X1F,	^X20,	^X20,	^X20

For $k = 0, 2, \dots, 32$, the k -th entry in the A_TABLE is the value $2^{(k/32)}$ rounded to 113 bits.

[illegible]

OTSSPOWGG
2-004

- G REAL*8 ** G REAL*8 power routine
DECLARATIONS

K 15

16-SEP-1984 01:58:59 VAX/VMS Macro V04-00
6-SEP-1984 11:28:13 [MTHRTL.SRC]OTSSPOWGG.MAR;1

Page 4
(2)

```
00000000 00000000 00000000 00004002 0280 170      .OCTA  ^X00000000000000000000000000004002
                                0290 171
                                09380828 61C5404A 0290 172 LOGTAB: .QUAD  ^X0938082861C5404A      : 0.13190956358802040E+02
                                3ED850C0 776C4052 0298 173      .QUAD  ^X3ED850C0776C4052      : 0.18466496512350972E+02
                                0410DC3A C709405E 02A0 174      .QUAD  ^X0410DC3AC709405E      : 0.30777494205631285E+02
                                00000000 00000000 02A8 175      .QUAD  ^X000000000000000000      : 0.00000000000000000E+00
                                00000004 02B0 176 LOGLEN = <.-LOGTAB>/8
                                02B0 177
636C74B5 78B0E177 B82F7652 71544007 02B0 178 C0:      .OCTA  ^X636C74B578B0E177B82F765271544007
                                02C0 179
                                02C0 180
                                FF06BA87 30963D64 02C0 181 EXPTAB: .QUAD  ^XFF06BA8730963D64      : 0.14345715681320842E-12
                                1AA0E6EF D8853DE5 02C8 182      .QUAD  ^X1AA0E6EFD8853DE5      : 0.39737266313176133E-10
                                7BD56FB9 B2AB3E63 02D0 183      .QUAD  ^X7BD56FB9B2AB3E63      : 0.91725627017354366E-08
                                80DDD703 6B083EDC 02D8 184      .QUAD  ^X80DDD7036B083EDC      : 0.16938509724215757E-05
                                C58FFF82 BFBD3F4E 02E0 185      .QUAD  ^XC58FFF82BFBD3F4E      : 0.23459619820224680E-03
                                39EFFF82 BFBD3F4E 02E8 186      .QUAD  ^X39EFFF82BFBD3F4E      : 0.21660849392498290E-01
                                00000000 00000000 02F0 187      .QUAD  ^X000000000000000000      : 0.00000000000000000E+00
                                00000007 02F8 188 EXPLEN = <.-EXPTAB>/8
                                02F8 189
```



```
07FC 02F8 249 .ENTRY OTSS$POWGG, ACMASK ;
      02FA 250
      02FA 251 :
      02FA 252 : Move x to R0. If x < 0, or x = 0 and y <= 0, return 'UNDEFINED
      02FA 253 : EXPONENTIATION' error condition, otherwise attempt to compute x**y
      02FA 254 :
      02FA 255 :
      7E 0C AC 56FD 02FA 256 CVTGH exp(AP), -(SP) ; Convert exp to h and put on stack
      50 04 AC 50FD 02FF 257 MOVG base(AP), R0 ; R0/R1 <-- x
      1A 14 0304 258 BGTR DEFINED ; If x > 0 attempt to compute x**y
      05 19 0306 259 BLSS UNDEFINED ; Branch to error code for x < 0
      6E B5 0308 260 TSTW (SP) ; Test sign of y
      01 15 030A 261 BLEQ UNDEFINED ; Branch to error condition if y <= 0
      030C 262
      030C 263 :
      030C 264 : If processing continues here, this implies that x = 0 and y > 0. Return
      030C 265 : with x**y = 0
      030C 266 :
      04 030C 267 RET ; Return
      030D 268
      030D 269 :
      030D 270 :
      030D 271 : If processing continues here, this implies that an undefined exponentiation
      030D 272 : was attempted. Signal error and return
      030D 273 :
      030D 274 :
      50 8000 8F 3C 030D 275 UNDEFINED:
      51 D4 0312 276 MOVZWL #X8000, R0 ;
      7E 00'8F 9A 0314 277 CLRL R1 ; R0/R1 <-- Reserved operand
      00000000'GF 01 FB 0318 278 MOVZBL #MTH$K UNDEXP, -(SP) ; Put error code on stack
      031F 279 CALLS #1, G^MTH$$SIGNAL ; Convert error number to 32 bit
      031F 280 ; condition code and signal error.
      031F 281 ; NOTE: Second argument is not re-
      04 031F 282 RET ; quired since there is no JSB entry.
      0320 283 ; Return
      0320 284
      0320 285 :
      0320 286 : If processing continues here will attempt to compute x**y as 2^[y*log2(x)].
      0320 287 : We begin by determining k and f such that x = 2^k*f, where 1 <= f < 2.
      0320 288 :
      0320 289 :
      58 50 FFFF800F 8F CB 0320 290 DEFINED:
      58 00004010 8F C2 0328 291 BICL3 #XFFF800F, R0, R8 ; R8 <-- 2^4*(biased exponent of x)
      50 58 C2 032F 292 SUBL #X4010, R8 ; R8 <-- 2^4*k=2^4*(exponent of x - 1)
      0332 293 SUBL R8, R0 ; R0/R1 <-- f = 2*(fraction field of x)
      0332 294
      0332 295 :
      0332 296 : We are now ready to compute log2(x). This computation is based on the
      0332 297 : following identity:
      0332 298 :
      0332 299 :
      0332 300 : 
$$\log_2(2^k * f) = k + \log_2(a) + \frac{2}{\ln(2)} \sum_{j=1}^{\infty} \frac{1}{2^j} z^{(2j+1)}, \text{ where } z = \frac{f-a}{f+a}.$$

      0332 301 :
      0332 302 :
      0332 303 : We begin by determining a as b^i, where b = 2^(1/32) and i is between 0
      0332 304 : and 32 inclusive. Specifically i is chosen by table look-up so that
      0332 305 : the magnitude of z is minimized. Since log2(a) = i/32, we may write
```

```

0332 306 :
0332 307 : log2(2^k*f) = k + i/32 + z*p(z^2).
0332 308 :
0332 309 : Note that in order to insure an accurate result, log2(2^k*f) must be computed
0332 310 : accurately to 68 bits. This will require some H-format arithmetic.
0332 311 :
0332 312 :
0332 313 EVAL_LOG2:
0332 314 CVTGH R0, R0 ; R0/R3 <-- f
0336 315 ROTL #7, R0, R4 ; Rotate low fraction bits of f
033A 316 BICL #^XFFFFFFF80, R4 ; R4 <-- index to INDEX table
0341 317 MOVW INDEX[R4], R4 ; R4 <-- i
0347 318 MOVAW (R4)[R8], R8 ; R8 <-- 2^5*(k + i/32)
034B 319 SUBH3 A_TABLE[R4], R0, R4 ; R4/R7 <-- f - a (NOTE: result is
0353 320 ; exact, i.e. no roundoff error)
0353 321 INCL R0 ; R0/R3 <-- 2*f
0355 322 SUBH2 R4, R0 ; R0/R3 <-- f + a
0359 323 DIVH3 R0, R4, -(SP) ; SP --> z
035E 324 :
035E 325 : Compute 2^5*z*p(z^2) = z*(c0 + c2*z^2 + c4*z^4 + c6*z^6)
035E 326 : = z*(c0 + q(z^2))
035E 327 :
035E 328 :
035E 329 :
035E 330 :
035E 331 CVTHG (SP), R6 ; R6/R7 <-- z (in G)
0362 332 MULG3 R6, R6, R0 ; R0/R1 <-- z^2
0367 333 POLYG R0, #LOGLEN-1, LOGTAB ; R0/R1 <-- q*(z^2)
036E 334 CVTGH R0, R0 ; R0/R3 <-- q*(z^2)
0372 335 ADDH2 C0, R0 ; R0/R3 <-- c0 + q*(z^2)
0378 336 MULH2 (SP)+, R0 ; R0/R3 <-- 2^5*z*p(z^2)
037C 337 :
037C 338 : Compute log2(x) = k + i/16 + z*p(z)
037C 339 :
037C 340 :
037C 341 :
037C 342 CVTLH R8, R4 ; Convert 2^5*(k + i/32) to H
0380 343 ADDH2 R4, R0 ; R0/R3 <-- 2^5*log2(x)
0384 344 :
0384 345 : We now compute x**y = 2^[y*log2(x)] by writing y*log2(x) as
0384 346 : y*log2(x) = I + j/32 + g/32,
0384 347 : where I is an integer, j is an integer between 0 and 31 inclusive, and
0384 348 : g is a fraction in the interval [-1/2, 1/2). It will be convenient to
0384 349 : make an initial check for overflow/underflow at this time
0384 350 :
0384 351 :
0384 352 :
0384 353 :
0384 354 :
0384 355 MULH2 (SP), R0 ; R0/R3 <-- 2^5*y*log2(x)
0388 356 BICW3 #^X8000, R0, R4 ; R4 <-- exp field of 2^5*y*log2(x)
038E 357 CMPW #^X4010, R4 ;
0393 358 BLSS EXCEPTION_1 ;
0395 359 CVTRHL R0, R8 ; R8 <-- 2^5*(I + j/32)
0399 360 CVTLH R8, R4 ;
039D 361 SUBH2 R4, R0 ; R0/R3 <-- 2^5*g
03A1 362 :

```



```

03A1 363 :
03A1 364 : We can now compute
03A1 365 :
03A1 366 :  $x**y = 2^{[y \cdot \log_2(x)]} = 2^{[I + j/32 + g/32]}$ 
03A1 367 :
03A1 368 :  $= (2^I) * [A * (B+1)] = 2^I * [A + A*B]$ , where
03A1 369 :
03A1 370 :  $A = 2^{(j/32)}$  is obtained from the A_TABLE and  $B = 2^{(g/32)} - 1$  is obtained
03A1 371 : by a Min/Max approximation
03A1 372 :
03A1 373 :
54 58 FF14 CF 50 50 76FD 03A1 374 CVTHG R0, R0 ; R0/R1 <-- 2^5*g
54 58 FCC6 CF 50 55FD 03A5 375 POLYG R0, #EXPLEN-1, EXPTAB ; R0/R1 <-- B = 2^(g/32) - 1. (Chebyshev
54 58 FFFFFFFE0 8F CB 03AC 376 ; polynomial approx. of degree 5)
54 58 FCC6 CF 44 7DFD 03AC 377 BICL3 #^FFFFFFE0, R8, R4 ; R4 <-- index into A_TABLE table
54 58 52 54 76FD 03B4 378 MOV0 A_TABLE[R4], R4 ; R4/R7 <-- A = 2^(j/32)
54 58 52 54 44FD 03BB 379 CVTHG R4, R2 ; R2/R3 <-- A
54 58 52 54 44FD 03BF 380 MULG2 R2, R0 ; R0 <-- A*B
54 58 50 50 56FD 03C3 381 CVTGH R0, R0
54 58 50 54 60FD 03C7 382 ADDH2 R4, R0 ; R0 <-- A*B + A
54 58 50 50 76FD 03CB 383 CVTHG R0, R0 ; R0 <-- 2^[(j + g)/32]
54 58 58 1F AA 03CF 384 BICW #^X1F, R8 ; R8 <-- 2^5*I
54 58 58 8F 9C 03D2 385 ROTL #-1, R8, R8 ; R8 <-- 2^4*I
54 58 50 58 A0 03D7 386 ADDW R8, R0 ; R0 <-- 2^[I+(j+g)/32]
54 58 0F 50 B1 03DA 387 CMPW R0, #^XF ; test for over/underflow
54 58 07 15 03DD 388 BLEQ EXCEPTION_2 ; see what exception is if neg or = 0
54 58 04 04 03DF 389 RETURN: RET ; otherwise return result in R0
03E0 390 :
03E0 391 :
03E0 392 : Handlers for software detected over/underflow conditions follow
03E0 393 :
03E0 394 :
03E0 395 EXCEPTION_1:
03E0 396 TSTF R0 ; if big ARG > 0 goto overflow
03E2 397 BGEQ OVER ; handler, otherwise go to
03E4 398 BRB UNDER ; underflow handler
03E6 399 EXCEPTION_2:
03E6 400 TSTW R8 ; test sign of I; if I < 0
03E8 401 BGEQ OVER ; go to overflow handler
03EA 402 :
03EA 403 :
03EA 404 : Underflow; if user has FU set, signal error. Always return 0.0
03EA 405 :
03EA 406 :
03EA 407 UNDER: CLRQ R0 ; R0 = result.
03EC 408 BBC #6, SF$W_SAVE_PSW(FP), 2$ ;
03F1 409 ; has user enabled floating underflow?
03F1 410 MOVZBL #MTH$K_FLOUNDMAT, -(SP) ; trap code for floating underflow.
03F5 411 ; Convert to MTH$ FLOUNDMAT
03F5 412 ; (32-bit VAX-11 exception code)
03F5 413 CALLS #1, G^MTH$$SIGNAL ; signal condition
03FC 414 2$: RET ; return
03FD 415 :
03FD 416 :
03FD 417 : Signal floating overflow, return reserved operand, -0.0
03FD 418 :
03FD 419 :
03FD 419 : else process for overflow

```


OTSS\$POWGG
2-004

C 16
- G REAL*8 ** G REAL*8 power routine 16-SEP-1984 01:58:59 VAX/VMS Macro V04-00 Page 9
OTSS\$POWGG - G REAL*8 to G REAL*8 giving 6-SEP-1984 11:28:13 [MTHRTL.SRC]OTSS\$POWGG.MAR;1 (4)

7E	00'8F	9A	03FD	420	OVER:	MOVZBL	#MTH\$K_FLOOVEMAT, -(SP)	; Put overflow code on stack
50	01	0F	79	0401		ASHQ	#15, #T, R0	; R0 = result = reserved operand -0.0.
				0405				; R0 will be copied to signal mechanism
				0405				; vector (CHF\$L_MCH_R0/R1) so it can be
				0405				; fixed up by any error handler
00000000'GF	01	FB	0405	425		CALLS	#1, G^MTH\$\$SIGNAL	; signal condition
		04	040C	426		RET		; return - R0 restored from CHF\$L_MCH_R0/R1
			040D	427				
			040D	428		.END		

OTSSPOWGG
Symbol table

D 16
- G REAL*8 ** G REAL*8 power routine

16-SEP-1984 01:58:59 VAX/VMS Macro V04-00
6-SEP-1984 11:28:13 [MTHRTL.SRC]OTSSPOWGG.MAR;1

Page 10
(4)

ACMASK	= 000007FC		
A TABLE	00000080	R	02
BASE	= 00000004		
CO	000002B0	R	02
DEFINED	00000320	R	02
EVAL_LOG2	00000332	R	02
EXCEPTION_1	000003E0	R	02
EXCEPTION_2	000003E6	R	02
EXP	= 0000000C		
EXPLEN	= 00000007		
EXPTAB	000002C0	R	02
INDEX	00000000	R	02
LOGLEN	= 00000004		
LOGTAB	00000290	R	02
MTH\$SIGNAL	*****	X	00
MTH\$K_FLOOVEMAT	*****	X	00
MTH\$K_FLOUNDMAT	*****	X	00
MTH\$K_UNDEXP	*****	X	00
OTSSPOWGG	000002F8	RG	02
OVER	000003FD	R	02
RETURN	000003DF	R	02
SFSW_SAVE_PSW	= 00000004		
UNDEFINED	0000030D	R	02
UNDER	000003EA	R	02

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_OTSSCODE	0000040D (1037.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	31	00:00:00.10	00:00:01.07
Command processing	107	00:00:00.76	00:00:04.73
Pass 1	127	00:00:02.23	00:00:10.32
Symbol table sort	0	00:00:00.04	00:00:00.06
Pass 2	88	00:00:01.12	00:00:04.75
Symbol table output	4	00:00:00.04	00:00:00.04
Psect synopsis output	2	00:00:00.01	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	361	00:00:04.31	00:00:21.02

The working set limit was 1050 pages.
9152 bytes (18 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 53 non-local and 1 local symbols.
488 source lines were read in Pass 1, producing 14 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macros defined

_S255\$DUA28:[SYSLIB]STARLET.MLB;2

4

88 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:OTSP0WGG/OBJ=OBJ\$:OTSP0WGG MSRC\$:MTHJACKET/UPDATE=(ENH\$:MTHJACKET)+MSRC

0264 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

